Highlight and Selection Control for Dynamic Table Visualization

Nada Golmie and Bill Kules
CMSC 838S Term Paper
University of Maryland
College Park, MD 20742
Email: {golmie, wmk}@cs.umd.edu

ABSTRACT
This paper describes two techniques to help solve the problem of viewing, navigating and analyzing medium size tables. We focus on column selection and highlight control and demonstrate the two techniques as part of a table viewer implemented in Java.

Keywords: Tables, Interactive data exploration, Overview + detail technique, Highlighting, Outliers.

INTRODUCTION
The advent of the World Wide Web makes access to electronic data easier and more effective than ever before. Both network capacity and the number of users have been growing at almost an exponential rate. Federal and state agencies produce an astounding variety of information that is of interest to the public at large, and they are expending a significant amount of effort to make that information available via the Web. Much of this information is in the form of tabular statistics. Unfortunately, the wide availability of statistical data does not mean the data is immediately useful. Most of the statistical information available on the web today is presented in a static tabular form that is hard for an untrained user to understand or manipulate. Users often have to scroll through several pages of a table display with thousands of entries and remain severely constrained in the way they can view and interact with the data.

These tables often contain many more columns of data than can be displayed on a single screen. One of the challenges a user faces when viewing these tables is to find the columns of interest, maintaining the context of the overall table while viewing the region that fits on the screen. We propose an overview + detail technique that provides a scaled-down view of all column headers for viewing and navigating medium to large tables (on the order of 50 x 50 cells) which are too large to display in their entirety on one screen. This provides a simple visual cue for the user to maintain context.

When viewing data, users often need to find basic information such as the maximum and minimum values and outliers. We additionally demonstrate a technique for highlighting outliers in tabular data based on a dynamic query concept that helps the user effectively interact with the data and spot outliers.

Our objective is to demonstrate a few techniques to help solve the problem of disseminating and viewing tabular data on the web. For this project, we assume that the table is computed or made available from a server, and transmitted to the user’s machine where it is viewed with little or no further interaction with the server. This is important to facilitate the rapid feedback required for browsing any spreadsheet.

The rest of this paper is structured as follows. In section 2, we provide an overview of tabular presentations and discuss their usage. Section 3 discusses related work and presents a literature survey on the topic. In section 4 we describe the details of our table viewer. In section 5 we conduct a comparative evaluation of our system with Excel and InfoZoom based on a task-based predictive model and number of mouse clicks. Section 6 contains general observations and suggested improvements based on early use of the tool. We summarize our contributions in section 7.

TABULAR DATA
A tabular display represents an important class of data visualization. It provides a compact and organized structure that lends itself to data analysis and comparison. It also facilitates understanding relations among different data elements. Wainer in [9] describes four purposes for table usage.

- exploration - tables help answer questions and understand the data.
- communication - tables provide effective means for presenting data. Each table has a story to tell.
- storage - archival data is generally stored in tables that would later aid in data retrieval.
- illustration - tables can be used in the same way graphs are used to accompany the text of an article or a presentation.

Although an effective tabular display can be a powerful tool in data presentation, there are a number of difficulties that need to be overcome in order to make the best use of it. Data presented statically is usually difficult to handle for large data
sets (more than a dozen objects). Another difficulty stems from having to scroll and thus lose context information such as column headers and row entries. In general the lack of interactive control that allows the user to manipulate (sort, compare) the data represents a significant challenge.

Thus building an effective tabular display is not a simple task. In addition to establishing the purpose of the tabular display, one needs to consider the audience or the intended users of the table. Similar to the classification of computer users given in [11], we categorize table users into three groups. Novice or first-time users have little or no knowledge of the information they are about to find in the table. Knowledgeable intermittent users are familiar with the table and know their task, but they do not always know how to accomplish it. Finally expert frequent users are “power users” who know their way around and whose main focus is to get the job done and done quickly.

Another factor to consider in building a tabular display (and this applies to other forms of display as well) is to understand the types of questions that will be asked of the data. Better answers can be found if questions are known a priori. It is almost impossible to discuss effective tabular display abstractly without considering the kind of questions tables can be used to answer. There are three levels of questions described by Wainer [10].

- Elementary questions typically involve single category data inspection or extraction.
- Intermediate questions involve comparisons, trends, and possibly relationships between different data entries.
- Overall questions require a thorough understanding of the data structure and the ability to build a level of abstraction for the entire data set.

RELATED WORK

Wainer in [9][10] gives three rules for improving a tabular presentation. He suggests (1) ordering the rows and columns in a way that makes sense, (2) rounding and using no more than two digits (this argument is based on human cognitive skills), and (3) providing an overall view by adding relevant summary information (such as means, maxima, and minima).

Marchionini in [1][2] provides much of the motivation for our project, as well as a framework for the larger problem into which our project fits. He describes the challenges organizations face when trying to provide access to statistical information to citizens. He characterizes the problem and describes an approach that combines metadata and user interface techniques at all levels to provide an integrated browser for statistical data.

The Table Lens [4] has influenced much of the recent work in table visualizations. It provides a number of user operations to view and sort tables. Another sophisticated table browser, FOCUS, is described in [3]. It provides a number of ways for users to explore a set of cases and variables. It uses a focus+context technique to permit viewing a large table of cases and attributes without scrollbars. It integrates several other techniques for sorting, grouping, and filtering.

Dynamic Queries [7] is an interactive technique that allows the user to manipulate sliders to control the amount of data displayed. In this project we use a button control technique very similar to the slider technique to display column headers.

Eick in [5] describes an innovative way to overload sliders with several kinds of context information. This concept sparked the idea for the column selection technique proposed in our project. We also use sliders to control highlights of columns entries.

MetricsView described in [6] is a Java applet that allows users to construct their own view of the information to be displayed in either tabular or graph format. It includes interface controls along the X/Y axes to interactively manipulate the dimensions mapped to the axes. It also provides for animations between the tabular and graph representations.

The Aggregate Manipulator framework [8] extends Dynamic Queries [7] and describes several techniques for aggregation and hierarchical ordering, controlling scope, focus of attention and level of details. Most of these techniques can be applied to tabular data.

In [22], IBM/Lotus developers demonstrate several tools that operate on spreadsheets to highlight cells based on simple statistical measures such as minimum, maximum and mean values. The user selects a range of cells, then applies the appropriate highlighting tool, which is configured via a property sheet.

Most commercial spreadsheets like Excel allow the user to selectively control the number of columns and rows displayed on the screen by providing a hide/unhide feature control.

HIGHLIGHT AND SELECTION CONTROL TECHNIQUES

The number of columns in a table that can be displayed on a screen is generally limited between 5 to 10 columns depending on the screen size. Solutions like TableLens [4] and InfoZoom [3] make use of zooming (factor 10:1) in order to display large databases in a table format. The user has to zoom in on an specific part of the table, and view it alongside a zoomed out view of the remaining part of the table. On the other hand, the need for displayed text to be legible defines another, more subtle boundary: if the size of text cannot be reduced below a threshold of legibility, then, as the need to display more columns increases, the user is required to increase the level of interaction with the table in order to get to the desired information.

The design of our tool, the Highlighted Dynamic Table Viewer (HiDTV) interface, balances the following two opposing forces: the need to display more columns on the screen (we’ll call...
this the "physical" limitation), and the need to keep information in context (we’ll call this the "functional" limitation). In addition navigational or functional controls must be added to this balance in order to allow user interaction with the information on-screen (selecting columns to be viewed, adding new columns, comparing column contents, inspecting outliers, etc.). Most spreadsheet packages such as Excel use filters that can be applied on a selected subset of the data (column/row) in order to view relations or perform various sorting, binning and thresholding operations. In this project we focus on one particular aspect of table browsing control and investigate the use of controlled highlighting to help the user quickly identify outliers in table entries. Until now most highlighting of electroning documents has only been used in a syntactic way to cause the viewer to notice that a phrase is "clickable" or "selected". In a recent version of LOTUS Smart Suite, a similar highlighting technique was used to apply the results of search queries on table entries (such as minimum and maximum).

These functional requirements suggest several improvements on the simple scrolling and data filtering techniques that are commonly encountered when viewing and interacting with large tables. We use a button bar control technique for maintaining context. This is based on an overview + detail concept that does not require any zooming as opposed to the focus + context used in [4][3]. For controlling highlighters we use a highlight slider based on a dynamic queries concept [7] for interactive data manipulation.

To evaluate the merits of our techniques, we implement HiDTV as a Java application. The window of the prototype is set to 700 x 400 pixels (resizable to any other size) and contains a control panel and a display region. The control panel occupies the top portion of the window and represents 15 % of the display area as shown in Figure 1.

**Button Bar Control**

A horizontal bar of toggle buttons is placed in a panel above the display region of the table to let the user select columns to display from the data table. Above this “button bar control” there is a label area. As the pointer moves over, or brushes, each button, the associated column header is displayed in the label area. The color of the buttons indicate the columns currently displayed. This provides a simple visual cue for the user to maintain context. The button bar contains a fixed number of buttons, one for each column in the data table. When a button is selected (darker gray color) the column that is associated with it in the underlying table is displayed in the view. The order of the columns in the display region corresponds to the order of buttons selected from left to right. This tight coupling between the control buttons and the columns in the view helps the user maintain context information for each column. Figure 2 shows a detail of the button bar control.

**Highlight Slider**

The highlighting slider gives the user a way to explore outliers. For each column containing numeric values, the column header contains a range slider in addition to the column label. Our range slider is similar to the double-edge sliders discussed by Eick [5], with two thumbs to define a range of values instead of a single point. The left and right ends of the slider correspond to the minimum and maximum values for the column. The two thumbs define three ranges of values, lower outliers on the left, the typical values in the middle, and upper outliers on the right. Every cell in the column with a value less than the left thumb is highlighted in green. The right thumb similarly defines the upper outliers, which are highlighted in red. The user can drag a thumb to adjust one endpoint of the range. Alternatively, the user can drag both thumbs simultaneously by grabbing the area between them. This allows the user to sweep through a range of values. Our intent is to provide a tight coupling between the slider and the display by repainting the column while the user adjusts the slider. Figure 3 shows a detail of the Highlight Sliders and several highlighted upper and lower outliers.

**Implementation**

The tool is currently a standalone Java application. It uses standard and custom Swing components. We have built the tool using both the 1.1 JDK with Swing and the 1.2 JDK.

The Swing table classes are used as the basis for our work. The standard TableModel is extended to support a mapping between the underlying data and the current view, controlled by the button bar.

The structure of the table classes using the Model-View-Controller paradigm allowed us to extend the table interface by "plugging in" our own components as needed.
The column headers required several such extensions. Since the standard column headers are display-only, they cannot contain a component such as the range slider. In Java parlance, they can render the header values, but not edit them. We augmented the header to do both. This allowed us to install a range slider at the top of each numeric column. The range slider implementation was taken from the Stardom project. The Editable Headers were adapted from anonymous examples found on the Web.

Each cell in the table is rendered using a custom renderer, which colors the text based on the current value of the range slider.

The tool uses a simple parser to import the ASCII data file. The first line of the data file has the column headers, and the rest of the file is the data. It automatically determines the data type (text, integer, floating point number) from the input for later use when highlighting outliers.

**COMPARATIVE EVALUATION**


**Example Data Set**

In order to illustrate our table view interface we choose a 1900 to 1990 urban and rural population census statistics available from the US Census Bureau web site (http://www.census.gov/population/censusdata/urpop0090.txt). Different definitions apply to the terms “urban” and “rural” depending on the year the census was conducted. The Census Bureau defines “urban” for the 1990 census as comprising all territory, population and housing units in places of 2,500 or more persons incorporated as cities, villages, boroughs (except in Alaska and New York) and towns (except in the six New England States, New York and Wisonin) but excluding the rural portions of extended cities. Territory, population and housing units not classified as urban constitute “rural”. The table consists of 50 columns including the total, the rural, and the urban population counts and percentages for a given year between 1900 and 1990. There were 66 row entries categorized by states or regions. The original table displayed on the web is split over several pages (15 columns per page) and occupies 6 printed pages using a small font.

**Methods**

Numerous methods to evaluate user interfaces have been investigated in the literature. These methods can be classified into two broad categories: methods requiring detailed task descriptions and methods that are task-independent. Methods like GOMS [16][15] require detailed task descriptions and take advantage of the interactive aspect of a user interface to measure its effectiveness in completing a specific set of tasks. On the other hand task-independent methods like the work of Tullis [18] use several metrics such as overall and local density, grouping and layout complexity to predict user performance. More recent research has focused on automating interface evaluation [19]. Shneiderman et al. in [13][14] consider aspect ratio, total number of widgets, non-widget area, widget density, margins, area balances, colors, and fonts to automatically check the design and consistency of dialogue boxes. If methods requiring detailed task descriptions are viewed as one end of a spectrum, automated evaluators can be placed at the other end. For the purpose of evaluating HiDTV, we use a hybrid system based on a task-based click model [20].
Metrics

One task independent metric that we use is **table compactness**. We define table compactness as the number of columns, and rows that can be displayed per unit area. Table compactness measures the efficient utilization of a limited display space. Although table compactness may depend on several factors such as column width, text fonts and may be difficult to accurately measure for different interface environments, we use it nevertheless to ensure that our techniques for column selection and highlight control do not add much overhead to the real estate of the tabular display.

In addition we use a **task-based click model** that counts the number of mouse clicks (including page downs and scrolls) similar to what is presented in [20]. This model attempts to estimate user’s effort in interacting with a table interface while performing the tasks described. Although this model does not predict the speed or the time a user requires for completing a task, it gives a rough estimate on performance time. The overall user effort is proportional to the number of clicks and keystrokes in addition to other cognitive parameters such as reading, comparing, and recalling words and executing mental steps. In [21] a similar click model is used to predict time performance and it is subsequently verified by experimentation. Thus measuring time performance often requires conducting an experimental study. Pirillo in [17] uses a form of GOMS analysis and time-cost estimates for perceptual and cognitive parameters to predict user time performance.

**Task Scenarios**

We use three representative task types of tabular data exploration in our evaluation. These tasks require both a local and a global search of the table [12] to assess special properties of the data presented and build relations among different row and column entries. We define a task taxonomy consisting of three categories.

- **Type 1** involves a simple table lookup and the ability to locate a single value in a relatively large data set of information. This corresponds to extracting a simple fact from the data and answering elementary questions [9]. In this case, it consists of associating a data point entry with a row and column.
- **Type 2** represents single column or row scanning tasks. This type requires a basic understanding of the data and the ability to find outliers (e.g. minimum, maximum values) and value ranges over a single column or row.
- **Type 3** focuses on the ability to perceive a trend in the table and to conduct a global search. It may require the user to look for changes across columns and rows, and to form a mental image of the data. In addition, the user may be required to understand how changes in one variable affect other variables.

Table 1 gives examples for the task taxonomy described.

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Task Definition</th>
<th>Task Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What was the urban population count for New York in 1920?</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>What is the region that has the maximum rural population count in 1990?</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Identify the regions that have a urban population count between 10,000,000 and 20,000,000 in 1990</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>What is the trend (increase/decrease) in the rural population count for the Northeast Region?</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>What is the difference between the Northeast and Midwest regions in terms of urban population count between 1950 and 1990?</td>
<td>3</td>
</tr>
</tbody>
</table>

**Table 1: List of Task Scenarios**

**Predictive Analysis**

We use the data set described above and the list of tasks given in Table 1 in order to measure compactness and the number of clicks required to complete a task. Screen snapshots for Excel and InfoZoom displaying the population census data are given in Figures 4 and 5 respectively. The compactness measurements are summarized in Table 2. The computation is based on counting the number of rows and columns displayed per unit area. We use the default font and display settings for Excel and InfoZoom and we do not use any special features for hiding columns or rows from the display. We also resize the column width in all three tools so that they have the same display semantics (i.e. text strings in all row and column entries including the headers appear in their entirety). We observe from Table 2 that both Excel and InfoZoom have approximately the same row and column compactness. Note that this result corresponds to a “zoomed out” mode in InfoZoom where all column headers are legible. HiDTV is slightly more compact where twice as many rows and columns fit within the same display area.

<table>
<thead>
<tr>
<th>HiDTV</th>
<th>Excel</th>
<th>InfoZoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.16 col. per in²</td>
<td>0.09 col. per in²</td>
<td>0.1 col. per in²</td>
</tr>
<tr>
<td>0.41 row per in²</td>
<td>0.26 row per in²</td>
<td>0.24 row per in²</td>
</tr>
</tbody>
</table>

**Table 2: Compactness Measurements**

The number of clicks required to perform a task is given in Table 3 for all three interfaces. We adjust the display size for all three interfaces so that to view the full table we would require 5 horizontal and 2 vertical scrolls. In HiDTV no horizontal scrolling is required unless the number of columns selected exceeds the display size. The calculations given in Table 3 depend on the item position in the row and column. For example, New York is located in the eleventh row so that no scroll down is required. Our calculations are based on worst case assumptions.
Table 3: Number of Mouse Clicks per Task

<table>
<thead>
<tr>
<th>Task</th>
<th>HiDTV</th>
<th>Excel</th>
<th>InfoZoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

Analysis of HiDTV

In HiDTV a type 1 task requires a total number of 4 clicks. A further analysis leads to the following breakdown: one click per column selection, and a maximum of two vertical scrolls in order to find the row corresponding to New York.

Type 2 tasks (corresponding to tasks 2 and 3) require 5 clicks. In this case we have one more click than in the type 1 case to adjust the highlight slider to show the maximum value.

Type 3 tasks require more clicks because we select more columns. There is also more (horizontal and vertical) scrolling involved in order to conduct comparisons between multiple rows and columns. For example, task 4 requires a total of 13 clicks: one click per column selected for a total of 10 columns for Region and population count between 1990 and 1900, 2 vertical scrolls to find the row, and an additional horizontal scroll to do a row scan.

Analysis of Excel and InfoZoom

Type 1 and type 2 tasks in Excel and InfoZoom require a total of 7 clicks. The detailed breakdown for each type could be slightly different depending on how tasks are performed. However they both add up to the same number of clicks. For type 1 a maximum of 5 horizontal and two vertical scrolls may be required in order to find the desired column and row respectively. For type 2 tasks, sorting filters can be applied to a selected column in order to find a maximum value or values within a specified range. So for tasks 2 and 3, we have 5 horizontal scrolls, a column selection, and a filter application (for example sort in descending order). Note that the trial version of InfoZoom that we use does not support a sorting filter on the column entries and we had to use scrolls instead. This leads us to believe that although the number of clicks in InfoZoom is the same as for Excel, more time would be required to perform a type 2 task in InfoZoom.

Type 3 tasks require a total of 12 and 10 clicks for tasks 4 and 5 respectively. For task 4, we count a maximum of 2 vertical scrolls to find the desired row, 9 clicks to hide unwanted columns or adjust the column display (to show only urban population count for each year from 1900-1990), and an extra horizontal scroll to do a row scan. For task 5, we count 2 vertical scrolls to find a row, 5 column adjustments (1950-1990), one horizontal scroll and additional 2 vertical scrolls to find the find the second region.

Table 3 shows that the number of clicks for HiDTV for type 1 and type 2 tasks that require single column and row scanning is less than for Excel and InfoZoom. Also we hypothesize that the time required to identify a set of outliers in HiDTV would be less than for the other two interfaces since it only requires adjusting the highlight slider. Also we note that the number of clicks for HiDTV is directly proportional to the number of columns selected. As the number of columns selected grows, the number of clicks required increases. Therefore for multiple row and column scan (type 3 tasks), using HiDTV may not constitute a definite advantage over Excel and InfoZoom.
OBSERVATIONS AND SUGGESTED IMPROVEMENTS

Initial use of the tool revealed a number of issues not previously identified, which in turn suggested refinements to the original concepts.

Button Bar

The small vertical dimension of the buttons makes it hard to sweep in one single gesture. The pointer “falls off” the bar and the user’s eye is forced to alternate rapidly between the pointer location and the current label. One possible solution would be to expand the button bar vertically. Another solution might be to use a Tool Tips approach and display the full column label at the pointer location. A third suggestion was to display more than one column label at a time above the button bar. This yielded the idea of using a scrolling marquee coupled to the location of the pointer when it is over the button bar. The marquee would contain all the column labels for the data. As the pointer swept across the buttons, the marquee would smoothly scroll in the appropriate direction. This would provide another form of context, and the smooth animation would avoid location confusion. Additionally, the column labels for the brushed column could be highlighted in the marquee and in the data table, providing another visual cue to the user.

This technique is targeted for columns, but it is not clear that it would be useful for rows. It might be useful to combine it with other techniques such as hierarchies of rows that can be collapsed (used in Excel and FOCUS).

One early idea not yet implemented would be to rotate column labels 90 degrees and use them as the labels in the buttons. The buttons would be taller, but there would be no need for the text box above the button bar. It would be interesting to see if enough of the label can be made visible to provide the appropriate overview.

Highlighter

The dynamic interaction of the range slider presents several challenges to the user in the current implementation. The usable range is limited to the width of the column. This makes it hard to precisely position a thumb. The colors of the three ranges in the slider should correspond to the highlight colors in the table and should be user-selectable. It might also be useful to display the specific values of range end points near the slider.

General Observations

These techniques focus on providing visible controls for manipulating views on a table. This makes the controls immediately apparent, but it does not significantly reduce the number of clicks to achieve the tasks. This implies that they may be useful for novice or intermittent users. These users need operations to be visible (since the users can’t be expected to remember a key sequence or menu location) and intuitive (since users will by definition not be familiar with the interface semantics). For these reasons, our tool might be simpler for the novice to use than similar features in Excel, Lotus, the Table Lens, or InfoZoom.

“Power users” would most likely find macros, keyboard shortcuts, or other techniques more powerful and more useful. Such techniques require more training and are more complex than our techniques, and they can do correspondingly more.

SUMMARY

We have demonstrated the use of a Java application for dynamic viewing and highlighting of tabular data. Our main contribution is the use of a button bar to control the selection of columns based on an overview + detail concept. We also explore a method of highlighting outliers with a range slider.

In this paper we report on our implementation and the concepts used in the design. Our interface focuses on two aspects of table browsing, namely, keeping information in context and spotting outliers.

We conduct a comparative evaluation of our prototype with two commercial products, Excel and InfoZoom. Using a simple mouse click model analysis we show that our interface requires less clicks to perform single column and row scan tasks. At this time we have tested the interface ourselves and believe that is has potential for time savings for novice and intermittent users. We would like to refine the concepts and conduct a user experiment in order to verify our hypotheses.

ACKNOWLEDGMENTS

We would like to acknowledge the help and suggestions provided by Dr. Ben Shneiderman in the design of this interface. We would also like to thank Dr. Michael Spenke from the German National Research Center for Information Technology (GMD), for making available a trial version of InfoZoom.

The Highlighting Sliders were adapted from the STARDOM project at the UMD-CP and described in [23]. The Editable Headers were adapted from examples posted by an anonymous author on the Web.

REFERENCES


22. Developments in Human-Computer Interaction, Programme 2 (Video), January 1997, IBM, Austin Texas.